Enc. (1)

(13)

AD-A166 728

CURITY CLASSIFICATION OF THIS PAGE (When Data Entered) UNCLASSIFIED

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
TITLE (and Subtitle) ROUND ROBIN SCHEDULING FOR FAIR FLOW CONTROL IN DATA COMMUNICATION NETWORKS		5. TYPE OF REPORT & PERIOD COVERED paper
		6. PERFORMING ORG. REPORT NUMBER LIDS-P-1537
AUTHOR(s) Ellen L. Hahne and Robert G. Gallager		8. CONTRACT OR GRANT NUMBER(s) DARPA Order No. 3045/2-2-84 Amendment #11 ONR/N00014-84-K-0357
PERFORMING ORGANIZATION NAME AND ADDRESS assachusetts Institute of Technology aboratory for Information and Decision Systems Cambridge, Massachusetts 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Program Code No. 5T10 ONR Identifying No. 049-383
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, Virginia 22209		12. REPORT DATE March 1986
		13. NUMBER OF PAGES 11
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Office of Naval Research Information Systems Program Code 437 Arlington, Virginia 22217		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release: distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Round robin link scheduling, in conjunction with conventional window flow control, can be used to achieve throughput fairness in point-to-point packet networks with virtual circuit routing.

86 4 15 002

March 1986 Report LIDS-P-1537

To be presented at the IEEE International Conference on Communications, June 1986

ROUND ROBIN SCHEDULING FOR FAIR FLOW CONTROL IN DATA COMMUNICATION NETWORKS

Ellen L. Hahne and Robert G. Gallager

Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

ABSTRACT

Round robin link scheduling, in conjunction with conventional window flow control, can be used to achieve throughput fairness in point-to-point packet networks with virtual circuit routing.

1. INTRODUCTION

Consider a data communication network consisting of store-and-forward nodes joined by point-to-point links. Each user session is assigned a fixed path (often called a virtual circuit) through the network, and data for the session are sent in packets along this path. In such a network it is possible for the incoming traffic rate at a node to exceed the outgoing rate, causing a data queue to build up at that node. This queue may eventually overflow the node's storage

This research was conducted at the M.I.T. Laboratory for Information and Decision Systems and was supported in part by the Defense Advanced Research Projects Agency under Contract ONR/N00014-84-K-0357 and by the National Science Foundation under Grant NSF-ECS-8310698. Ellen Hahne was also partially supported by an AT&T Bell Laboratories Scholarship.

space, or the delay of acknowledgments may cause transmitters to assume that data were lost. These problems result in wasteful retransmissions that effectively reduce the capacity of the network. Flow control procedures attempt to prevent or alleviate this degradation by regulating the appropriate traffic sources. Reference 1 discusses many of the flow control techniques that have been proposed in the literature.

One such scheme is the *window method* [1]. This technique limits the number of packets for each session which have been transmitted but for which acknowledgments have not yet been received. The maximum permissible number of outstanding packets is called the *window size*. A single window may be applied to all of a session's traffic, or the session may have a separate window for its traffic over each link. We mention the window method because it is a component of several more elaborate strategies to be discussed later.

It would be desirable for flow control procedures to regulate network inputs so as to grant each session a fair throughput rate. As explained in Reference 1, many proposed flow control methods are rather unfair. Several studies have, however, addressed the fairness issue, and we will briefly discuss these now.

The problem of achieving throughput fairness can be broken down into three parts. First the fairness objective must be formulated precisely. Then the fair session throughputs must be determined. Finally, these rates must be enforced. The objective of Gallager and Golestaani [2] is to minimize a sum of penalty functions, one for each link and one for each session. The link functions penalize high link delays, while the session functions penalize low session throughputs. This objective function expresses a trade-off between overall network efficiency and user fairness. Another fairness criterion, called *max-min flow*, is used in various forms by Bially, Gold, and Seneff [3], Jaffe [4], Hayden [5], Gafni and Bertsekas [6], and Mosely [7]. We will define only the simplest version of this objective, which is Hayden's. To satisfy the max-min flow criterion, the smallest session rate in the network must be as large as possible. Subject to this constraint, the second-smallest session rate must be as large as possible, etc. Given a network with its link capacities and a set of sessions with their routes, there is a unique set of session rates that satisfies the max-min conditions. Section 2 explains the max-min flow criterion in more detail. We will adopt this criterion as the definition of fairness for this paper. The studies mentioned in this paragraph also develop distributed algorithms for computing session rates that are fair according to the various criteria.

Once the desired session rates are computed, there are several ways to enforce them. Hayden [5] and Mosely [7] simulate a session input control that produces packet lengths proportional to the desired session rate. The time between packet admissions is approximately constant. This control is particularly meaningful for packetized voice traffic: it represents the output of a variable rate vocoder [3]. Bially, Gold, and Seneff [3] simulate a similar control. Another possibility is to use fixed length packets, but to regulate the time between packet admissions for each session. Mukherji [8] does this in a way that is less rigid than time-division multiplexing and thus avoids the delay problems of TDM under light loads. A third approach, studied by Gallager and Golestaani [2], uses window flow control and adjusts the sessions' window sizes to achieve the desired rates.

In this paper we propose another strategy for achieving max-min fair session rates. Let each link offer its packet transmission slots to its user sessions in round robin fashion. If a session is offered a chance to use a link slot but has no packets ready, then that same slot is offered to the next session, and perhaps the next, etc., until a ready session is found. In each pass of a link's round robin, a session may transmit only one packet. In order to prevent an excessively long queue at a session's bottleneck link, window flow control is also employed. Under certain simplifying assumptions, it can be shown that this strategy yields long-term average session throughputs that are max-min fair. This and related results are covered in Section 4. Section 3 gives the assumptions underlying the results. The most noteworthy assumptions are that all sessions have heavy demand and that the window sizes are sufficiently large.

The attraction of this method is its simplicity. Note that the fair rates are never explicitly computed, as they are for other fair flow control schemes. The only overhead communication is that required for the window acknowledgments. The window sizes do not need to be adjusted as network conditions change. A potential practical problem with this method is that the windows may need to be large in order to guarantee throughput fairness for some networks. This point is discussed in Section 5. Section 6 summarizes and concludes the paper.

2. FAIRNESS CRITERION

This section describes the simplest version of the max-min flow criterion, which we will take as our definition of throughput fairness. First, let us describe the network flow model in terms of which the criterion is defined. The network consists of nodes joined by directed links. Two nodes may be connected by any number of links in either or both directions. The links have finite capacity. The topology of the network and the link capacities are given. A set of n one-way communication sessions $x_1, \cdots, x_n$ has been specified, and each session has been assigned a path (i.e., a sequence of appropriately directed links) through the network. The goal is to assign a feasible transmission rate r_j to each session x_j so as to treat sessions fairly. It is assumed that the traffic for each session will form a continuous, steady flow at the assigned rate.

Next, let us define some terms. An *allocation* $R = (r_1, \cdots, r_n)$ specifies a non-negative real rate r_j for each session x_j without violating the link capacities; that is, the sum of the rates for all sessions sharing any particular link cannot exceed the link's capacity. The *rate list* of an allocation $R = (r_1, \cdots, r_n)$ is the nondecreasing permutation of R. Note that the elements of a rate list are not necessarily distinct.

Now fairness can be defined. An allocation R satisfies the *max-min flow criterion* if no other allocation has a rate list that is lexicographically greater than the rate list of R. In other words, the smallest component rate of R is as large as possible and, subject to that constraint, the second-smallest component rate of R is as large as possible, etc. Each of these nested optimization problems can be formulated as a linear program [5], and it can be shown that there exists a unique allocation that satisfies them all.

This max-min allocation will be called the *fair* allocation, because it can be shown to be the only allocation with the following property: a session x_j cannot transmit above its assigned rate r_j unless some session x_k with assigned rate $r_k \leq r_j$ transmits below its assigned rate.

Alternatively, the max-min flow criterion can be stated in terms of bottlenecks. Suppose some allocation is given. A link l is called a *bottleneck* for a session x_j using l if the assigned rate r_j of x_j is at least as large as the assigned rate of any other session using l, and if the entire capacity of l is assigned to the sessions using it. It can be shown that an allocation satisfies the max-min flow criterion if and only if every session has at least one bottleneck link.

These concepts will now be illustrated for the system in Figure 1.

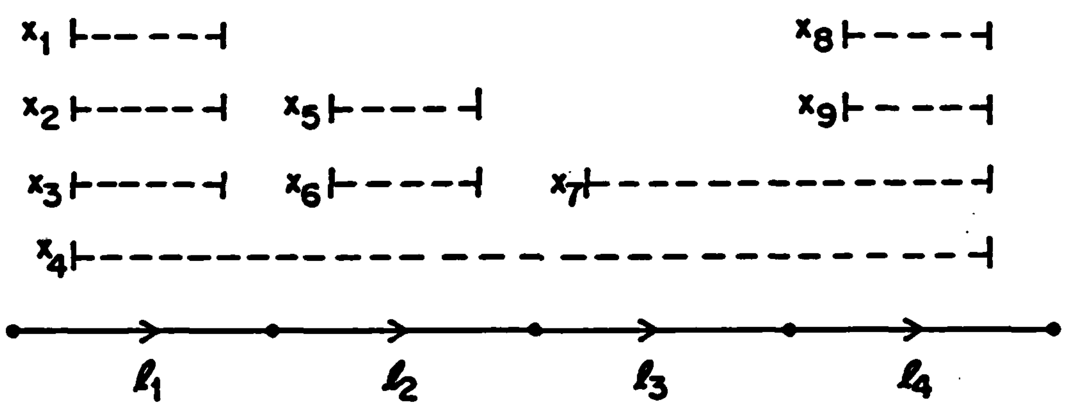

FIGURE 1

The network consists of links l_1, l_2, l_3, and l_4 in tandem. Each link has unit capacity. Sessions x_1, x_2 and x_3 use only link l_1. Session x_4 uses all four links. Sessions x_5 and x_6 use only link l_2. Session x_7 uses l_3 followed by l_4. Sessions x_8 and x_9 use only l_4. The max-min allocation is (1/4, 1/4, 1/4, 1/4, 3/8, 3/8, 1/4, 1/4, 1/4). The rate list for this allocation is (1/4, 1/4, 1/4, 1/4, 1/4, 1/4, 1/4, 3/8, 3/8). Sessions x_1, x_2, x_3, and x_4 have link l_1 as a bottleneck. Sessions x_5 and x_6 have l_2 as a bottleneck. Sessions x_4, x_7, x_8, and x_9 have l_4 as a bottleneck; note that x_4 is bottlenecked at both l_1 and l_4. Link l_3 is not a bottleneck for any session, since it has unused capacity.

3. SYSTEM MODEL

This section presents the system model assumed in Section 4. This model is slightly less general than that of Section 2 with regard to network topology and link capacities. However, the main difference between the two models is that Section 2 assumes a continuous, steady traffic flow for each session, whereas this section explicitly models packets, queues, windows, and schedules.

In this detailed model, the network consists of store-and-forward nodes joined by point-to-point, one-way communication links. If two nodes are connected by link(s) in one direction, then they must be connected by at least one link in the reverse direction so that flow control acknowledgments can be returned. Links and nodes are error-free and perfectly reliable. The storage capacity of each node is large enough that overflow is impossible.

All links have the same capacity, and all data packets have the same length. A packet experiences no processing delay at a node, other than a possible queuing delay as it waits for transmission. A packet experiences no propagation delay on a link. The packet transmission slots of all links are synchronized. Thus the entire system operates with slotted time.

Each session consists of a one-way flow of data packets from some origin node to some destination node. Several sessions may have the same origin and destination nodes. During the time interval in which the system is analyzed, the set of sessions using the network is fixed. Each session is assigned a path (i.e., a sequence of appropriately directed links) through the network. It is assumed that a session always has packets waiting at its origin node and has storage available at its destination node. Since the sessions and their routes are given and fixed, the max-min fair session rates of Section 2 are well-defined and do not change over time. (Of course, it is not clear at this point whether the average packet flows of the sessions will actually match these ideal rates.) The bottleneck link(s) for each session are also well-defined in terms of the max-min rates, as explained in Section 2.

Each session has a buffer at each node along its path that can store up to W packets waiting for transmission. Associated with each buffer are W logical quantities called *permits*. Each packet waiting in a buffer must hold a permit for that particular buffer. Permits for a buffer that are not currently held by packets in that buffer are stored at the node immediately upstream. When a packet is transmitted from the n^{th} node of its path to the $(n+1)^{st}$ node, it relinquishes the permit it needed for waiting at node n, and it seizes a permit for waiting at node $n+1$. (If no such permits are available at node n, the packet cannot be transmitted.) During the same time slot in which the packet, carrying its new permit, is transmitted from node n to node $n+1$, the relinquished permit is transmitted from node n back to node $n-1$ over some link in this reverse direction. This discipline guarantees that the buffers will never overflow. It is known as *link-by-link window flow control*, with W as the window size. The link capacity consumed by the overhead communication required to implement permits will be ignored.

Each link l has a round robin scheduler to decide which session will use the link during each time slot. The scheduler at l consults a fixed data structure consisting of session identifiers arranged in a directed ring. Each session using l appears exactly once on this ring. The scheduler also maintains a variable called the *ring position* identifying the session that last sent a packet over l. To allocate the current time slot, the scheduler searches the ring, starting with the session immediately following the current ring position, until it finds the first session x that has both packet(s) and permit(s) available. If there is such a session x, it transmits a packet over l during the current slot, and the ring position is updated to x.

4. RESULTS

Consider a system that satisfies the assumptions of Section 3. Suppose that the following parameters of the system are given: the network topology, the set of sessions using the network, the sessions' paths, the window size W, and the round robin ring for each link.

Suppose that the following initial conditions are also given: the queue length for each session at each link and the ring position of the scheduler at each link. Let L denote the number of links in the network. Let H denote the maximum number of links in the path of any session in the network. Let S denote the maximum number of sessions sharing any single link in the network. Define Δ_1 and Δ_2 in terms of L, H, S, and W as follows:

$$\Delta_1 = 2\,H^L\,S^L$$

$$\Delta_2 = 3\,H^L\,S^{2L}\,W$$

The results below hold for this system, provided that the window size is larger than Δ_1 packets.

- The long-term average throughput of each session exactly equals its max-min fair rate.
- The number of packets transmitted for any session over any link during any time interval is within Δ_2 packets of the max-min fair amount, regardless of the length of the time interval.
- There exists a time T such that the following statements are true *after* T:
  + The number of packets transmitted for any session over any link during any time interval is within Δ_1 packets of the max-min fair amount, regardless of the length of the time interval. Recall that Δ_1 is independent of the window size W.
  + Every session has at least one bottleneck link, called a *pure bottleneck*, where there are always packets and permits waiting, i.e., where the session accepts every chance offered to it by the round robin link scheduler. A link that is a pure bottleneck for some session is a pure bottleneck for every session bottlenecked there.
  + The range (i.e., maximum minus minimum) of the queue length for any given session at any given link is at most Δ_1 packets.
  + The lengths of a session's queues are related to the locations of its bottleneck links. Buffers that are "slightly" upstream of bottleneck links are sometimes full and are never empty; buffers that are "slightly" downstream of bottleneck links

are sometimes empty and are never full. To make this claim precise, we define buffer properties P_E , P_N and P_F below.

P_E : The buffer is empty infinitely often and is never full.
P_N : The buffer is never empty and is never full.
P_F : The buffer is full infinitely often and is never empty.

We will now characterize the buffers of a given session with respect to properties P_E , P_N and P_F . All buffers upstream of the session's first bottleneck link satisfy property P_F . All buffers downstream of the last bottleneck link satisfy P_E . The set of buffers between two successive bottleneck links can be partitioned into three (possibly empty) subsets, with the buffers in each subset being contiguous. Buffers in the upstream subset satisfy P_E . Buffers in the downstream subset satisfy P_F . The middle subset can contain at most one buffer, which must satisfy P_N .

The claims above can be proved by induction, starting with those sessions having the smallest max-min fair rate, then considering those sessions with the second-smallest fair rate, etc. The proof is given in [9].

5. REMARKS ON THE WINDOW SIZE

Recall that in Section 4 the window size is assumed to be larger than $\Delta_1 = 2 H^L S^L$. For all but the simplest networks, this quantity is impractically large. Large windows could result in substantial storage requirements, high cross-network delay, very bursty session flows, and slow convergence of the average session throughputs to the max-min fair rates. An important question is whether a large window is actually necessary to guarantee max-min fair session throughputs. The answer is complicated, because the exact window size needed for perfect fairness in a particular system depends strongly on the network topology, the set of sessions and their routes, the order of the sessions in the round robin rings, and the initial queue lengths throughout the network. Unfortunately, examples have been discovered for which a very large window is, in fact, necessary to exactly achieve the max-min fair rates. It is not known how "common" such examples are. The natural next question is this: for a given small window size, how unfair can the session throughputs be? We are currently investigating this issue.

6. CONCLUSION

Round robin link scheduling, in conjunction with conventional window flow control, can be used to achieve throughput fairness in point-to-point packet networks with virtual circuit routing. Assuming heavy demand and large flow control windows, it can be proved that the long-term average throughputs of the sessions are fair, in the sense of maximizing the minimum session rate. The round-robin method is considerably simpler than some other strategies for throughput fairness. The performance of round robin scheduling with smaller windows and lesser, more random demand deserves further study.

REFERENCES

[1] Gerla, M. and L. Kleinrock, "Flow Control: A Comparative Survey", *IEEE Trans. Comm.*, Vol. COM-28, No. 4, April 1980, pp. 553-574.

[2] Gallager, R. G. and S. J. Golestaani. "Flow Control and Routing Algorithms for Data Networks", *Proc. Fifth Internatl. Conf. on Comp. Comm.*, Oct. 1980. pp.779-784.

[3] Bially, T., B. Gold. and S. Seneff, "A Technique for Adaptive Voice Flow Control in Integrated Packet Networks", *IEEE Trans. Comm.*, Vol. COM-28, No. 3, March 1980, pp. 325-333.

[4] Jaffe, J. M., "A Decentralized, 'Optimal'. Multiple-User. Flow Control Algorithm", *Proc. Fifth Internatl. Conf. on Comp. Comm.*, Oct. 1980, pp. 839-844.

[5] Hayden, H. P., "Voice Flow Control in Integrated Packet Networks", Report LIDS-TH-1152, Lab. for Info. and Decision Sys., Mass. Inst. of Technology. Cambridge, Mass., Oct. 1981.

[6] Gafni. E. M. and D. P. Bertsekas, "Dynamic Control of Session Input Rates in Communication Networks", *IEEE Trans. Auto. Control*, Vol. AC-29, No. 11, Nov. 1984, pp. 1009-1016.

[7] Mosely, J., "Asynchronous Distributed Flow Control Algorithms", Report LIDS-TH-1415, Lab. for Info. and Decision Sys., Mass. Inst. of Technology, Cambridge, Mass., Oct. 1984.

[8] Mukherji, U., "A Schedule-Based Approach for Flow-Control in Data Communication Networks", Report LIDS-TH-1527, Lab. for Info. and Decision Sys., Mass. Inst. of Technology, Cambridge, Mass., Jan. 1986.

[9] Hahne, E. L., "Round Robin Scheduling for Fair Flow Control in Data Communication Networks", Ph.D. Thesis, Dept. of Elec. Engr. and Comp. Sci., Mass. Inst. of Technology, Cambridge, Mass., forthcoming.

Distribution List

Defense Documentation Center Cameron Station Alexandria, Virginia 22314	12 Copies
Assistant Chief for Technology Office of Naval Research, Code 200 Arlington, Virginia 22217	1 Copy
Office of Naval Research Information Systems Program Code 437 Arlington, Virginia 22217	2 Copies
Office of Naval Research Branch Office, Boston 495 Summer Street Boston, Massachusetts 02210	1 Copy
Office of Naval Research Branch Office, Chicago 536 South Clark Street Chicago, Illinois 60605	1 Copy
Office of Naval Research Branch Office, Pasadena 1030 East Greet Street Pasadena, California 91106	1 Copy
Naval Research Laboratory Technical Information Division, Code 2627 Washington, D.C. 20375	6 Copies
Dr. A. L. Slafkosky Scientific Advisor Commandant of the Marine Corps (Code RD-1) Washington, D.C. 20380	1 Copy

Office of Naval Research — 1 Copy
Code 455
Arlington, Virginia 22217

Office of Naval Research — 1 Copy
Code 458
Arlington, Virginia 22217

Naval Electronics Laboratory Center — 1 Copy
Advanced Software Technology Division
Code 5200
San Diego, California 92152

Mr. E. H. Gleissner — 1 Copy
Naval Ship Research & Development Center
Computation and Mathematics Department
Bethesda, Maryland 20084

Captain Grace M. Hopper — 1 Copy
Naval Data Automation Command
Code OOH
Washington Navy Yard
Washington, DC 20374

Advanced Research Projects Agency — 1 Copy
Information Processing Techniques
1400 Wilson Boulevard
Arlington, Virginia 22209

Dr. Stuart L. Brodsky — 1 Copy
Office of Naval Research
Code 432
Arlington, Virginia 22217

Prof. Fouad A. Tobagi
Computer Systems Laboratory
Stanford Electronics Laboratories
Department of Electrical Engineering
Stanford University
Stanford, CA 94305